IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	Allen	Docket No.:	ST-99-AD-037
Filed:	December 14, 2001	Examiner:	Chery, Mardochee
Serial No.:	10/016,972	Art Unit:	2188

For: Disk Controller Providing for the Auto-Transfer of Host-Requested-Data From a Cache Memory Within a Disk Memory System

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

This Appeal Brief is respectfully submitted in connection with the above-identified application in response to the Notice of Panel Decision from Pre-Appeal Brief Review mailed July 16, 2008. A Pre-Appeal Brief Request for Review was filed June 9, 2008.

REAL PARTY OF INTEREST (37 C.F.R. 41.37(c)(1)(i))

The present application is assigned to ST Microelectronics, Inc., the real party in interest.

RELATED APPEALS AND INTERFERENCES (37 C.F.R. 41.37(c)(1)(ii))

Appellant is not aware of any related appeals or interferences.

STATUS OF CLAIMS (37 C.F.R. 41.37(c)(1)(iii))

Claims 1-9 and 27-30 stand finally rejected. No claims have been allowed. Claims 10-26 are cancelled. Claims 1-9 and 27-30 are the subject of this appeal. The claims on appeal are reproduced in the attached Appendix.

STATUS OF AMENDMENTS (37 C.F.R. 41.37(c)(1)(iv))

No Amendments under 37 CFR §1.116 have been entered after final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R. 41.37(c)(1)(v))

Claim 1:

The invention of claim 1 relates to a mass storage system for a host system, the mass storage system having a mass storage device, a cache memory, a microprocessor, and a controller (all as addressed in more detail in the following paragraphs). While mass storage systems with cache memories to improve overall performance are known, conventional mass storage systems with cache memories slow down considerably when a portion of requested data is in the cache memory and a portion of the requested data is not in the cache memory. *Page 1, paragraph 0005.* The conventional mass storage systems resort to invoking a microprocessor to initiate a transfer of the portion of the requested data not in the cache memory to a host system after looking at other portions of the cache memory. Additionally, the transfer of the portion of the requested data not in the cache memory to the host system typically occurs sequentially with the transfer of the portion of the requested data in the cache memory to the host system. Significant costs are incurred in the use of the microprocessor to initiate the transfer of the portion of the requested data not in the cache memory after looking at other areas of the cache memory, including wasting valuable microprocessor processing time that may be used it other operations. *Page 1, paragraph 0005.*

Claim 1 provides for a mass storage system that initiates the auto-transfer of requested data from the cache memory irrespective of where in the cache memory either the whole of, or a portion of, the requested data resides. *Page 1, paragraph 0008*. Furthermore, when only a portion of the requested data resides in cache memory, the microprocessor is enabled to obtain the portion of the requested data not in the cache memory from the mass storage device, while the portion of the requested data in the cache memory is concurrently auto-transferred to the host system from the cache memory by operation of the disk-controller. *Page 1, paragraph 0009*. In this way, the disk memory system of the invention performs the auto-transfer without microprocessor intervention, thus leaving the microprocessor free to perform other functions, such as the retrieval of other data from the storage device. As a result, data is rapidly transferred to the host system, and performance of the microprocessor is improved. *Page 1, paragraph 0013 and page 6, paragraph 0108.*

Figure 1, reproduced below for convenience, illustrates an exemplary system connected to a host system 150, the system includes a mass storage device 140, a cache memory 120, a microprocessor 130, and a controller 110. The controller 110 includes a first circuit 111 to determine if at least a portion of requested data resides in the cache memory 120, a second circuit 112 to generate cache memory 120 access pointer addresses and values for the portion of the requested data residing in the cache memory 120, and an auto-transfer mechanism 113 to initiate an auto-transfer of the portion of the requested data residing in the cache memory 120 to the host system 150. *Page 3, paragraphs 0038 through 0041*. When the first circuit 111 detects that the entirety of the requested data is not in cache memory 120, the controller 110 invokes the microprocessor 130 to transfer the missing data from the mass storage device 140 to the cache memory 120 and then to a host system 150, concurrently with the requested data being auto-transferred to the host system 150 from the cache memory 120 by operation of the auto-transfer mechanism 113. *Page 3, paragraph 0043*. The controller 110 performs the auto-transfer without

the intervention of the microprocessor 130. The microprocessor 130 is therefore free to retrieve other data from the storage device 140, as needed. *Page 3, paragraph 0044*. When the first circuit 111 does not detect at least a portion of the requested data in the cache memory 120, microprocessor 130 operates to retrieve the requested data from the mass storage device 140. *Page 3, paragraph 0045*. Various other features of the invention are not necessary for an understanding of the issues on appeal and hence are not presented herein.

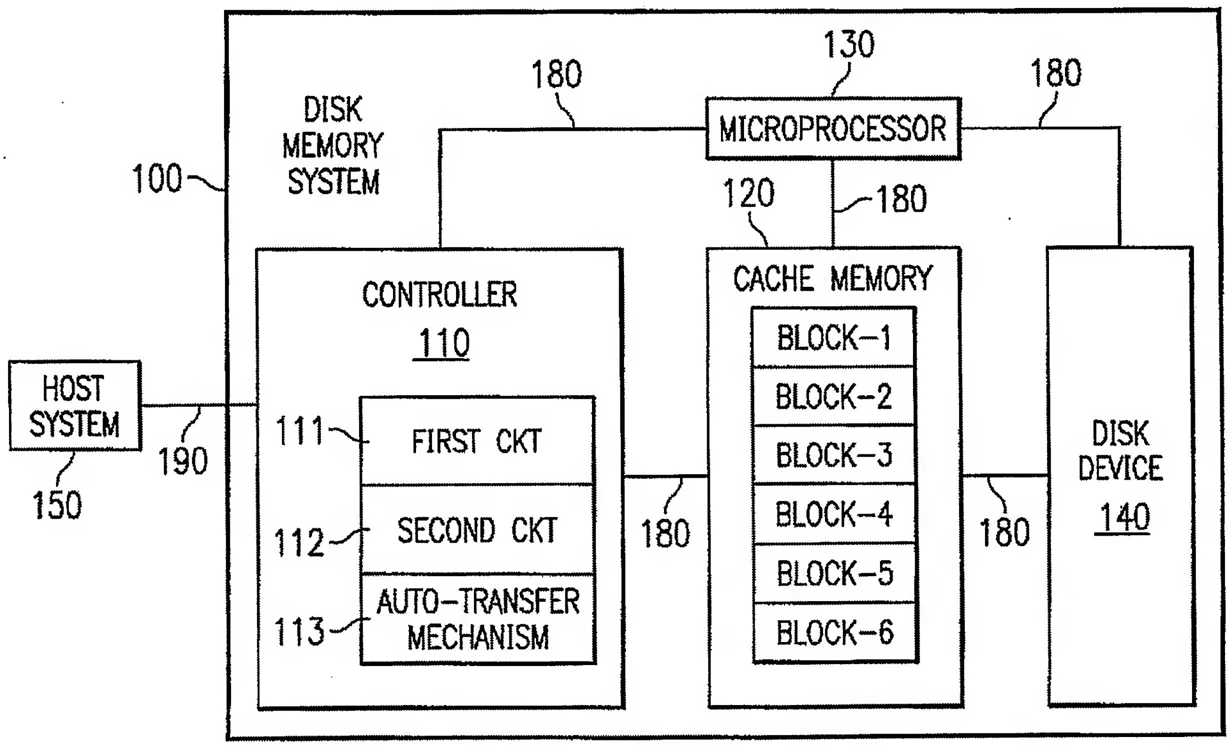

FIG. 1

Claim 8:

The invention of claim 8 relates to a method of retrieving data from a mass storage system and provides for a system that i) receives a data request from a host system, ii) if none of the requested data is in a cache memory, initiating a transfer of the requested data from the mass storage device, iii) if a portion of the requested data is in the cache memory and a portion of the

requested data is in the mass storage device, transferring the portion of the requested data from the cache memory to the host system substantially concurrently with transferring the portion of the requested data from the mass storage device directly to the host system, and iv) if all of the requested data is in the cache memory, transferring the requested data from the cache memory to the host system. *Page 6, paragraphs 0097 through 0102.*

Figure 10 illustrates a process or method flow start showing the operation of the mass storage system. In operation, the controller 110 determines if a first data block of the requested data is equal to a first data block in cache memory 120. *Page 6, paragraph 0096.* If yes, then the controller 110 determines if the cache memory 120 contains enough data blocks to satisfy the requested data, i.e., does all of the requested data reside in the cache memory 120. *Page 6, paragraph 0097.* If the cache memory 120 contains all of the requested data, then the controller 110 initiates the auto-transfer of the requested data from the cache memory 120. *Page 6, paragraph 0098.* If the cache memory does not contain all of the requested data, the controller 110 now operates to concurrently enable the initiation of the auto-transfer of the portion of the requested data residing in the cache memory 120 and invokes the microprocessor 130 to obtain the remaining portion of the requested data from the mass storage device 140. *Page 6, paragraph 0099.* Various other features of the invention are not necessary for an understanding of the issues on appeal and hence are not presented herein.

Claim 27:

The invention of claim 27 relates to a disk memory system connected to a host system, the disk memory system having a disk-device, a cache, a disk controller, a microprocessor, and logic means in the disk controller. Figure 1 illustrates an exemplary system, which includes a mass storage device 140, a cache memory 120, a microprocessor 130, and a controller 110. The logic

means in the controller 110 includes but is not limited to a first circuit 111 to determine if at least a portion of requested data resides in the cache memory 120, a second circuit 112 to generate cache memory 120 access pointer addresses and values for the portion of the requested data residing in the cache memory 120, and an auto-transfer mechanism 113 to initiate an auto-transfer of the portion of the requested data residing in the cache memory 120. *Page 3, paragraphs 0038 through 0041.* When the logic means (i.e., the first circuit 111) detects that the entirety of the requested data is not in cache memory 120, the controller 110 invokes the microprocessor 130 to transfer the missing data from the mass storage device 140 to the cache memory 120 and then to a host system 150, concurrently with the requested data being auto-transferred to the host system 150 from the cache memory 120 by operation of the logic means (i.e., the auto-transfer mechanism 113). *Page 3, paragraph 0043.* The controller 110 performs the auto-transfer without the intervention of the microprocessor 130. The microprocessor 130 is therefore free to retrieve other data from the storage device 140, as needed. *Page 3, paragraph 0044.* When the logic means (i.e., the first circuit 111) does not detect at least a portion of the requested data in the cache memory 120, microprocessor 130 operates to retrieve the requested data from the mass storage device 140. *Page 3, paragraph 0045.* Various other features of the invention are not necessary for an understanding of the issues on appeal and hence are not presented herein.

Claim 28:

The invention of claim 28 relates to a disk memory system connected to a host system, the disk memory system having a disk-device, a cache, a disk controller, and a microprocessor. Figure 1 illustrates an exemplary system, which includes a mass storage device 140, a cache memory 120, a microprocessor 130, and a controller 110. The controller 110 includes a first circuit 111 to determine if at least a portion of requested data resides in the cache memory 120, a second circuit

112 to generate cache memory 120 access pointer addresses and values for the portion of the requested data residing in the cache memory 120, and an auto-transfer mechanism 113 to initiate an auto-transfer of the portion of the requested data residing in the cache memory 120. *Page 3, paragraphs 0038 through 0041*. When the controller 110 detects that the entirety of the requested data is not in cache memory 120, the controller 110 invokes the microprocessor 130 to transfer the missing data from the mass storage device 140 to the cache memory 120 and then to a host system 150, concurrently with the requested data being auto-transferred to the host system 150 from the cache memory 120 by operation of the controller 110. *Page 3, paragraph 0043*. The controller 110 performs the auto-transfer without the intervention of the microprocessor 130. The microprocessor 130 is therefore free to retrieve other data from the storage device 140, as needed. *Page 3, paragraph 0044*. When the controller 110 does not detect at least a portion of the requested data in the cache memory 120, microprocessor 130 operates to retrieve the requested data from the mass storage device 140. *Page 3, paragraph 0045*. Various other features of the invention are not necessary for an understanding of the issues on appeal and hence are not presented herein.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL (37 C.F.R. 41.37(c)(1)(vi))

(1) Claims 1-6 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,696,931 to Lum, et al. ("Lum") over U.S. Patent No. 6,092,149 to Hicken, et al. ("Hicken") and further in view of U.S. Patent No. 6,301,605 to Napolitano, et al. ("Napolitano").

(2) Claims 8, 27-28, and 30 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Lum over U.S. Patent No. 6,141,728 to Simionescu, et al. ("Simionescu") and further in view of Napolitano.

(3) Claim 7 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Lum in view of Hicken, Napolitano, and of well-known practices in the art.

(4) Claim 29 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Lum in view of Simionescu, Napolitano, and of well-known practices in the art.

(5) Claim 9 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Lum in view of Simionescu, Napolitano, and U.S. Patent Publication No. 200110014929 A1 to Taroda, et al. ("Taroda").

ARGUMENT (37 C.F.R. 41.37(c)(1)(vii))

It is respectfully submitted that claims 1-9 and 27-30 recite patentable subject matter under the provisions of 35 U.S.C. § 103.

Rejection under 35 U.S.C. 103(a) as being unpatentable over Lum over Hicken and further in view of Napolitano

Claims 1-7:

The Examiner finally rejected claims 1-7 under 35 U.S.C. § 103 as being unpatentable over Lum over Hicken and further in view of Napolitano. The Examiner has also finally rejected claim 7 under 35 U.S.C. § 103 as being unpatentable over Lum over Hicken, Napolitano and well-known practices in the art.

Argument:

Claim 1 requires that the controller "initiates an auto-transfer of the requested data that resides in the cache to the host system; and requests a transfer of the requested data that resides in

the mass storage device directly to the host system" and that "wherein the request of the transfer and the initiation of the auto-transfer occur substantially concurrently" (emphasis added). Simply, the cited claim element requires that the transferring of the requested data from the cache to the host system occur directly and (1) transferring requested data from the cache to the host, (2) transferring requested data from mass storage directly to the host, and (3) steps (1) and (2) occur substantially concurrently.

Examiner has conceded that Lum does not teach transferring a portion of requested data from cache to host system substantially concurrently with transferring a portion of requested data from a mass storage device to the host.[1] In the past, Examiner has sought to overcome the conceded shortcomings of Lum by asserting that Hicken discloses the concurrent or substantially concurrent request of the transfer and the initiation of the auto-transfer.[2]

However, the very portion relied on by Examiner reveals Hicken's teachings are contrary to Appellant's claimed invention. Rather than transferring requested data residing in the mass storage device to the host concurrently with transferring requested data residing in the cache to the host, Examiner asserts that Hicken teaches maximizing performance through the execution of internal processes in parallel which include receiving a plurality of commands from the host including a read command for cached data and transferring data from the cache to the host.[3]

Rather than teaching the concurrent transfer of requested data from the cache memory to the host and from the mass storage device to the host, Hicken teaches the concurrent execution of commands by a host.[4] The commands include a read command for cached data and transferring data from the cache to the host. Appellant has reviewed Hicken and finds no teachings related to the

[1] See Final Office Action dated February 7, 2008, pg. 8.

[2] See Final Office Action dated April 20, 2007, pg. 2.

[3] See Final Office Action dated April 20, 2007, pg. 2.

[4] Hicken, column 2, lines 40-41.

concurrent transfer of requested data from the cache memory to the host and from the mass storage device to the host. Likewise, Examiner has failed to anywhere identify such teaching or even a suggestion of such teaching in Hicken.

Furthermore, a review of Office Actions subsequent to Final Office Action dated April 20, 2007 shows that Examiner has failed to maintain the rejection on claim 1's requirement for concurrency using Hicken. It would appear that Examiner has conceded the point that Hicken fails to teach or suggest this limitation. In fact, in the most recent rejection of claim 1, provided in the Final Office Action dated February 7, 2008, Examiner has nowhere identified in any of the asserted references a teaching or suggestion regarding the concurrent transfer of requested data from the cache memory to the host and from the mass storage device to the host. Hence, Examiner has failed to establish even a *prima facie* case of obviousness with regard to claim 1, and by dependency claims 2 - 7.

Claim 7 depends on claim 1 and further requires that "the mass storage system and the host system are integrated into a single unit." Claim 7 was rejected on substantially the same grounds as was claim 1. As addressed above with respect to claim 1, Examiner has conceded Lum does not teach transferring data-blocks from a cache to a host concurrently with transferring data blocks from a disk drive to said host. Furthermore, as explained in detail above, neither Hicken nor Napolitano teach the direct transfer of requested data directly from mass storage to host or that such transfers would be concurrently with transfers from a cache to the host. For at least these reasons, claim 7 is also patentably distinct over the prior art.

<u>Rejection under 35 U.S.C. 103(a) as being unpatentable over Lum over Simionescu and further in view of Napolitano</u>

Claims 8, 9, 27- 30:

The Examiner finally rejected claims 8, 27-28, and 30 under 35 U.S.C. § 103 as being unpatentable over Lum over Simionescu and further in view of Napolitano. The Examiner has also finally rejected claim 29 under 35 U.S.C. § 103 as being unpatentable over Lum over Simionescu, Napolitano, and well-known practices in the art, as well as claim 9 under 35 U.S.C. § 103 as being unpatentable over Lum in view of Simionescu, Napolitano, and Taroda.

Argument:

Claim 8 requires that "if a portion of the requested data is in the cache memory and a portion of the requested data is in the mass storage device, transferring the portion of the requested data from the cache memory to the host system <u>substantially concurrently</u> with transferring the portion of the requested data from the mass storage device <u>directly to the host system</u>" (emphasis added). Simply, the cited claim element requires (1) transferring requested data from the cache to the host, (2) transferring requested data from mass storage <u>directly</u> to the host, and (3) that steps (1) and (2) occur substantially concurrently.

Examiner has conceded that Lum does not teach transferring a portion of requested data from cache to host system substantially concurrently with transferring a portion of requested data from a mass storage device to the host.[5] To overcome the conceded shortcoming of Lum, Examiner has asserted that Simionescu discloses transferring a portion of the requested data from the cache memory to the host system substantially concurrently with transferring a portion of the requested data from the mass storage devices to the host to quickly and efficiently buffer multiple data transfers into the cache buffer.

[5] See Final Office Action dated February 7, 2008, pg. 8.

However, the very portion relied upon by Examiner reveals Simionescu's teachings are contrary to Appellant's claimed invention. Rather than transferring data that resides in the mass storage device directly to the host system, Simionescu teaches transferring data that resides in the mass storage device to the cache.

Further, rather than transferring data from the cache to the host substantially concurrently with the transfer of data from mass storage to the host, Simionescu merely teaches "At the same time, firmware causes the disk drive to read additional sectors from disk into the cache buffer, and these additional sectors are located during rescanning and are thereupon automatically transferred to the host."[6]

In other words, Simionescu's additional sectors (data) are moved from the disk drive into the cache (a first phase) and are required to wait in the cache and are not moved to the host (a second phase) until they are located during a subsequent rescanning. Only after they are located during the subsequent rescanning are they transferred to the host. Therefore, only after the subsequent rescanning of the cache, which requires additional time, are Simionescu's additional sectors located and transferred to the host. This is neither "directly to the host system" nor is it "substantially concurrently with the transfer of data from the cache to the host system."

While Appellants acknowledge that transferring data from a mass storage device to the host may entail some level of buffering, the two-step process described by Simionescu – in which data is transferred from the host to the cache and then subsequently (only after a subsequent rescanning process) transferred from the cache to the host – is a fundamentally different approach; one that does not render obvious claim 8.

---

[6] Simionescu, col. 21, lines 22-26.

Claim 9 is dependent on claim 8 and further requires that "the data request has a first logical address protocol and the cache memory has a second logical address protocol and including the step of translating between the first and second address protocols."

Claim 9 was rejected on substantially the same grounds as was claim 8. As addressed above with respect to claim 8, Examiner has conceded Lum does not teach transferring data-blocks from a cache to a host concurrently with transferring data blocks from a disk drive to said host. Furthermore, as explained in detail above, Simionescu neither teaches data transfers directly from a disk drive to a host nor that such transfers would be concurrent with transfers from a cache to the host. For at least these reasons, claim 9 is also patentably distinct over the prior art.

Claim 27 requires that "said logic means being operable to determine when a cache-hit-portion of data-blocks corresponding to said data-request reside in said cache and a cache-miss-portion of said data-blocks corresponding to said data-request do not reside in said cache, and operating in response to such a determination to concurrently cause said disk-controller to auto-transfer said cache-hit-portion of said data-blocks corresponding to said data-request from said cache, and to cause said microprocessor to fetch data-blocks corresponding to said cache-miss-portion of said data-request directly from said disk-device" (emphasis added). Likewise, claim 28 requires that "said logic circuit being operable to determine a partial-cache-hit when a first-portion of data-blocks corresponding to said data-request reside in said cache and a second-portion of said data-blocks corresponding to said data-request do not reside in said cache, and operating in response to a partial-cache-hit to concurrently cause said disk-controller to auto-transfer said first-portion of said data-blocks corresponding to said data-request from said cache, and to cause said microprocessor to fetch data-blocks corresponding to said second-portion of said data-request directly from said disk-device" (emphasis added).

Claim 29 depends on claim 28 and further requires that "the slow disk-device, the fast cache, the disk controller, and the microprocessor all of which are operationally and electrically interconnected to form a unitary disk memory system that appears as a single source of data-blocks to a host-system."

Claims 27, 28, and 29 were rejected on substantially the same grounds as was claim 8. As addressed above with respect to claim 8, Examiner has conceded Lum does not teach transferring data-blocks from a cache to a host concurrently with transferring data blocks from a disk drive to said host. Furthermore, as explained in detail above, Simionescu neither teaches data transfers directly from a disk drive to a host nor that such transfers would be concurrently with transfers from a cache to the host. For at least these reasons, claims 27, 28, and 29 are also patentably distinct over the prior art.

CONCLUSION

For the foregoing reasons, Appellant respectfully submits that the Examiner's final rejection of claims 1-9 and 27-30 under 35 U.S.C. § 103 is improper and respectfully requests that the Board of Patent Appeals and Interference so find and reverse the Examiner's rejections.

Respectfully submitted,

18 AUGUST 2008
Date

Steven H. Slater
Attorney for Appellant
Reg. No. 35,361

Slater & Matsil, L.L.P.
17950 Preston Road, Suite 1000
Dallas, TX 75252
ph: (972) 732-1001
fax: (972) 732-9218

CLAIMS APPENDIX

APPEALED CLAIMS

1. A mass storage system comprising:

a mass storage device;

a cache memory coupled to the mass storage device, the cache memory being organized in data blocks and having a first data block;

a microprocessor coupled to the mass storage device and the cache memory; and

a controller coupled to the microprocessor and the cache memory, wherein the controller:

receives a data request from a host system;

calculates new cache counter and pointer values when the first requested data block is not contained within the first block of the cache;

initiates an auto-transfer of the requested data that resides in the cache to the host system; and

requests a transfer of the requested data that resides in the mass storage device directly to the host system, wherein the request of the transfer and the initiation of the auto-transfer occurs substantially concurrently.

2. The mass storage system of claim 1 further comprising a controller register including:

a counter register containing a value for the number of blocks of data in the cache memory,

a start address register identifying the first block of data in the cache memory; and

a pointer register containing a pointer to the first block of data in the cache memory.

3. The mass storage system of claim 1 wherein the microprocessor transfers the requested data that resides in the mass storage device to the host system by way of the cache memory.

4. The mass storage system of claim 1 wherein the microprocessor controls the transfer of requested data that resides in the mass storage device and the controller controls the transfer of requested data that resides in the cache.

5. The mass storage system of claim 1 wherein the controller includes a general or special purpose processor executing program instructions.

6. The mass storage system of claim 1 wherein the transfer of requested data that resides in the mass storage device occurs substantially simultaneously with the transfer of data that resides in the cache.

7. The mass storage system of claim 1 wherein the mass storage system and the host system are integrated into a single unit.

8. A method of retrieving data from a mass storage system comprising:

receiving a data request from a host system, the data request including a block address for a first block of the requested data and a number of blocks in the request;

if none of the requested data is in a cache memory, initiating a transfer of the requested data from a mass storage device;

if a portion of the requested data is in the cache memory and a portion of the requested data is in the mass storage device, transferring the portion of the requested data from the cache memory to the host system substantially concurrently with transferring the portion of the requested data from the mass storage device directly to the host system;

if all the requested data is in the cache memory, transferring the requested data from the cache memory to the host system;

wherein the steps of transferring the requested data from the cache memory system include calculating a starting location in the cache memory for the transfer, based upon the block address and the number of blocks in the request received from the host system.

9. The method of claim 8 wherein the data request has a first logical address protocol and the cache memory has a second logical address protocol and including the step of translating between the first and second address protocols.

27. A disk memory system, comprising:

a disk-device for storing data-blocks on disk-storage-media;

a cache for storing data-blocks;

a disk-controller;

registers within said disk-controller containing a cache-start-address of a first data-block in said cache, and a cache-block-length that defines a total number of data-blocks stored in said cache;

said disk-controller receiving a data-request that contains a request-start-address of a first data-block in said data-request, and a request-block-length that defines a total number of data-blocks in said data-request;

a microprocessor operationally interconnecting said disk-device, said cache, and said disk-controller;

logic means in said disk-controller responsive to said cache-start-address as compared to said request-start-address, and to said cache- block-length-as compared to said request-block-length;

said logic means being operable to determine when no data-blocks corresponding to said data-request reside in said cache, and operating in response to such a determination to cause said microprocessor to fetch said data-blocks corresponding to said data-request from said disk-device;

said logic means being operable to determine when all of the data-blocks corresponding to said data-request reside in said cache, and operating in response to such a determination to cause said disk-controller to auto-transfer all of said data-blocks corresponding to said data-request from said cache without requiring operation of said microprocessor; and

said logic means being operable to determine when a cache-hit-portion of data-blocks corresponding to said data-request reside in said cache and a cache-miss-portion of said data-blocks corresponding to said data-request do not reside in said cache, and operating in response to such a determination to concurrently cause said disk-controller to auto-transfer said cache-hit-portion of said data-blocks corresponding to said data-request from said cache, and to cause said microprocessor to fetch data-blocks corresponding to said cache-miss-portion of said data-request directly from said disk-device.

28. A disk memory system, comprising:

a relatively slow disk-device for storing data-blocks on disk-storage-media;

a relatively fast cache for storing data-blocks;

a disk-controller, and a microprocessor;

registers within said disk-controller containing a cache-start-address of a first data-block in said cache, and a cache-block-length that defines a total number of data-blocks stored in said cache;

said disk-controller receiving as input a data-request from said host-system;

said data request containing a request-start-address of a first data-block in said data-request, and a request-block-length that defines a total number of data-blocks in said data-request;

a logic circuit in said disk-controller responsive to said cache-start address as compared to said request-start-address, and to said cache-block-length as compared to said request-block-length; said logic circuit being operable to determine a cache-miss when no data-blocks corresponding to said data-request reside in said cache, and operating in response to a cache-miss to cause said microprocessor to fetch said data-blocks corresponding to said data-request from said disk-device;

said logic circuit being operable to determine a total-cache-hit when all of the data-blocks corresponding to said data-request reside in said cache, and operating in response to a total-cache-hit to cause said disk-controller to auto-transfer all of said data-blocks corresponding to said data-request from said cache without requiring operation of said microprocessor; and

said logic circuit being operable to determine a partial-cache-hit when a first-portion of data-blocks corresponding to said data-request reside in said cache and a second-portion of said data-blocks corresponding to said data-request do not reside in said cache, and operating in response to a partial-cache-hit to concurrently cause said disk-controller to auto-transfer said first-portion of said data-blocks corresponding to said data-request from said cache, and to cause said microprocessor to fetch data-blocks corresponding to said second-portion of said data-request directly from said disk-device.

29. The disk memory system of claim 28 wherein the slow disk-device, the fast cache, the disk controller, and the microprocessor all of which are operationally and electrically

interconnected to form a unitary disk memory system that appears as a single source of data-blocks to a host-system.

30. The disk memory stem of claim 28 wherein the logic means include a processor executing programmed instructions.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.